[SREES] Seminarski - SystemModel Dokumentacija

Generated by Doxygen 1.9.4

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 SystemModel Namespace Reference

Classes

- class Bus
- class SystemModel

Typedefs

- using fi = std::pair< std::function< double(std::vector< double >)>, std::function< double(std::vector< double >)> >
- using dfidx = std::pair< std::vector< std::function< double(std::vector< double >)> >, std::vector< std↩ ::function< double(std::vector< double >)> > >
- using AdmittanceMatrix = std::vector< std::tuple< uint8_t, uint8_t, std::complex< double > > >
- using Branch = std::tuple< TypeOfBranch, uint8_t, uint8_t, double, double, double, double >

Enumerations

- enum class TypeOfBus { Slack , PV , PQ }
- enum class ThreePhaseLoadConfigurationsType { Star , GroundedStar , Delta }
- enum class TypeOfBranch { Line , Transformer }

Functions

- std::ostream & operator<< (std::ostream &stream, const SystemModel &systemModel)

  Output stream operator overload

4.1.1 Typedef Documentation

4.1.1.1 AdmittanceMatrix

```
using SystemModel::AdmittanceMatrix = typedef std::vector<std::tuple<uint8_t, uint8_t, std↩
::complex<double> >>
```

4.1.1.2 Branch

```
using SystemModel::Branch = typedef std::tuple<TypeOfBranch, uint8_t, uint8_t, double, double,
double, double>
```

4.1.1.3 dfidx

```
using SystemModel::dfidx = typedef std::pair<std::vector<std::function<double(std::vector<double>)>
>, std::vector<std::function<double(std::vector<double>)> >>
```

4.1.1.4 fi

```
using SystemModel::fi = typedef std::pair<std::function<double(std::vector<double>)>, std↩
::function<double(std::vector<double>)> >
```

4.1.2 Enumeration Type Documentation

4.1.2.1 ThreePhaseLoadConfigurationsType

```
enum class SystemModel::ThreePhaseLoadConfigurationsType [strong]
```

Enumerator

Star	
GroundedStar	
Delta	

4.1.2.2 TypeOfBranch

```
enum class SystemModel::TypeOfBranch [strong]
```

Enumerator

Line	
Transformer	

4.1.2.3 TypeOfBus

```
enum class SystemModel::TypeOfBus  [strong]
```

Enumerator

Slack	
PV	
PQ	

4.1.3 Function Documentation

4.1.3.1 operator<<()

```
std::ostream & SystemModel::operator<< (
            std::ostream & stream,
            const SystemModel & systemModel )
```

Output stream operator overload

Parameters

stream	Output stream object
systemModel	SystemModel object to be printed to the stream

Returns

Chapter 5

Class Documentation

5.1 SystemModel::Bus Class Reference

```
#include <systemModel.h>
```

Public Member Functions

- Bus (TypeOfBus typeOfBus)
- TypeOfBus getTypeOfBus () const
- void setVoltageMagnitude (double voltageMagnitude)

  Sets the value at which the voltage amplitude for the given bus should be maintained.
- void setVoltagePhase (double voltagePhase)

  Sets the value at which the voltage phase for the given bus should be maintained.
- void setActivePower (double activePower)

  Sets the value at which the active power for the given bus should be maintained.
- void setReactivePower (double reactivePower)

  Sets the value at which the rective power for the given bus should be maintained.
- std::optional< double > getVoltageMagnitude () const

  Gets the value at which the voltage magnitude for the given bus should be maintained.
- std::optional< double > getVoltagePhase () const

  Gets the value at which the voltage phase for the given bus should be maintained.
- std::optional< double > getActivePower () const

  Gets the value at which the active power for the given bus should be maintained.
- std::optional< double > getReactivePower () const

  Gets the value at which the rective power for the given bus should be maintained.

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Bus()

```
SystemModel::Bus::Bus (
            TypeOfBus typeOfBus )  [inline]
```

5.1.2 Member Function Documentation

5.1.2.1 getActivePower()

```
std::optional< double > SystemModel::Bus::getActivePower ( ) const
```

Gets the value at which the active power for the given bus should be maintained.

Returns

Value of active power for the bus

5.1.2.2 getReactivePower()

```
std::optional< double > SystemModel::Bus::getReactivePower ( ) const
```

Gets the value at which the rective power for the given bus should be maintained.

Returns

Value of reactive power for the bus

5.1.2.3 getTypeOfBus()

```
TypeOfBus SystemModel::Bus::getTypeOfBus ( ) const [inline]
```

5.1.2.4 getVoltageMagnitude()

```
std::optional< double > SystemModel::Bus::getVoltageMagnitude ( ) const
```

Gets the value at which the voltage magnitude for the given bus should be maintained.

Returns

Value of voltage magnitude of the bus

5.1.2.5 getVoltagePhase()

```
std::optional< double > SystemModel::Bus::getVoltagePhase ( ) const
```

Gets the value at which the voltage phase for the given bus should be maintained.

Returns

Value of voltage phase of the bus

5.1.2.6 setActivePower()

```
void SystemModel::Bus::setActivePower (
             double activePower )
```

Sets the value at which the active power for the given bus should be maintained.

Parameters

activePower	Value of active power for the bus

5.1.2.7 setReactivePower()

```
void SystemModel::Bus::setReactivePower (
             double reactivePower )
```

Sets the value at which the rective power for the given bus should be maintained.

Parameters

reactivePower	Value of reactive power for the bus

5.1.2.8 setVoltageMagnitude()

```
void SystemModel::Bus::setVoltageMagnitude (
             double voltageMagnitude )
```

Sets the value at which the voltage amplitude for the given bus should be maintained.

Parameters

voltageMagnitude	Value of voltage magnitude of the bus

5.1.2.9 setVoltagePhase()

```
void SystemModel::Bus::setVoltagePhase (
             double voltagePhase )
```

Sets the value at which the voltage phase for the given bus should be maintained.

Parameters

voltagePhase	Value of voltage phase of the bus

The documentation for this class was generated from the following files:

- systemModel.h
- systemModel.cpp

5.2 SystemModel::SystemModel Class Reference

```
#include <systemModel.h>
```

Public Member Functions

- SystemModel (uint8_t maxNumberOfBuses)
- AdmittanceMatrix getAdmittanceMatrix () const
- uint8_t getNumberOfBuses () const
- Bus & getBus (uint8_t busNumber)

  Gets the bus with the given bus number
- void addBus (TypeOfBus typeOfBus)

  Adds a bus to the system
- void addLoad (uint8_t busNumber, double activePower, double reactivePower)

  Adds a load to a bus
- void addLine (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b)

  Adds a line between buses
- void addGenerator (uint8_t busNumber, double voltageMagnitude, double activePower)

  Adds a generator to a bus
- void addSlackGenerator (uint8_t busNumber, double voltageMagnitude, double voltagePhase)

  Adds a generator to the slack bus
- bool hasSlackBeenAssigned () const

  Check whether the slack bus has been assigned
- void addTransformer (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g, double b)

  Adds a transformer between buses
- void addCapacitorBank (uint8_t busNumber, double b, ThreePhaseLoadConfigurationsType configuration↩ Type)

  Adds a capacitor bank to a bus
- fi getBusFunctions (uint8_t busNumber) const

  Gets the bus functions (fi_P and fi_Q) for the desired bus
- dfidx getDerivativesOfBusFunctions (uint8_t busNumber) const

  Gets the derivates of the bus functions (dfi_P/dx and dfi_Q/dx) for the desired bus (two rows of the Jacobian associated with the given bus functions)
- void removeBranch (uint8_t busNumber1, uint8_t busNumber2)

  Removes a line or transformer between buses
- void changeLine (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b)

  Changes the parameters of the line between buses
- void changeTransformer (uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g, double b)

  Changes the parameters of the transformer between buses
- std::vector< Branch > getBranches () const
- void removeBus (uint8_t busNumber)

  Removes the given bus from the system
- void removeCapacitorBank (uint8_t busNumber)

  Removes the capacitor bank that is connected to the given bus
- void changeCapacitorBank (uint8_t busNumber, double b, ThreePhaseLoadConfigurationsType configuration↩ Type)

  Changes the parameters of the capacitor bank connected to the given bus
- std::vector< std::tuple< uint8_t, double, ThreePhaseLoadConfigurationsType > > getCapacitorBanks () const

Friends

- std::ostream & operator<< (std::ostream &stream, const SystemModel &systemModel)

5.2.1 Constructor & Destructor Documentation

5.2.1.1 SystemModel()

```
SystemModel::SystemModel::SystemModel (
             uint8_t maxNumberOfBuses )  [inline]
```

5.2.2 Member Function Documentation

5.2.2.1 addBus()

```
void SystemModel::SystemModel::addBus (
             TypeOfBus typeOfBus )
```

Adds a bus to the system

Parameters

typeOfBus	Type of the bus (Slack, PV, PQ) to be added to the system

5.2.2.2 addCapacitorBank()

```
void SystemModel::SystemModel::addCapacitorBank (
             uint8_t busNumber,
             double b,
             ThreePhaseLoadConfigurationsType configurationType )
```

Adds a capacitor bank to a bus

Parameters

busNumber	Ordinal number of the desired bus
b	One phase susceptance of the bank
configurationType	Three phase load configuration type (delta, star, grounded star) of the bank

5.2.2.3 addGenerator()

```
void SystemModel::SystemModel::addGenerator (
            uint8_t busNumber,
            double voltageMagnitude,
            double activePower )
```

Adds a generator to a bus

Parameters

busNumber	Ordinal number of the desired bus
voltageMagnitude	Voltage magnitude on which the given bus should be maintained

<param name="activePower"Active power on which the given bus should be maintained>

5.2.2.4 addLine()

```
void SystemModel::SystemModel::addLine (
            uint8_t busNumber1,
            uint8_t busNumber2,
            double r,
            double x,
            double b )
```

Adds a line between buses

Parameters

busNumber1	Ordinal number of the first bus
busNumber2	Ordinal number of the second bus
r	Series resistance of the transmission line PI equivalent
x	Series reactance of the transmission line PI equivalent
b	Shunt susceptance of the transmission line PI equivalent

5.2.2.5 addLoad()

```
void SystemModel::SystemModel::addLoad (
            uint8_t busNumber,
            double activePower,
            double reactivePower )
```

Adds a load to a bus

Parameters

busNumber	Ordinal number of the desired bus
activePower	Active power drawn by the load
reactivePower	Reactive power drawn by the load

5.2.2.6 addSlackGenerator()

```
void SystemModel::SystemModel::addSlackGenerator (
            uint8_t busNumber,
            double voltageMagnitude,
            double voltagePhase )
```

Adds a generator to the slack bus

Parameters

busNumber	Ordinal number of the desired bus
voltageMagnitude	Voltage magnitude on which the given bus should be maintained
voltagePhase	Voltage phase on which the given bus should be maintained

5.2.2.7 addTransformer()

```
void SystemModel::SystemModel::addTransformer (
            uint8_t busNumber1,
            uint8_t busNumber2,
            double r,
            double x,
            double g,
            double b )
```

Adds a transformer between buses

Parameters

busNumber1	0rdinal number of the first bus
busNumber2	0rdinal number of the second bus
r	Series resistance of the transformer PI equivalent
x	Series reactance of the transformer PI equivalent
g	Shunt conductance of the transformer PI equivalent
b	Shunt susceptance of the transformer PI equivalent

5.2.2.8 changeCapacitorBank()

```
void SystemModel::SystemModel::changeCapacitorBank (
            uint8_t busNumber,
            double b,
            ThreePhaseLoadConfigurationsType configurationType )
```

Changes the parameters of the capacitor bank connected to the given bus

Parameters

busNumber	Ordinal number of the desired bus
b	One phase susceptance of the bank
configurationType	Three phase load configuration type (delta, star, grounded star) of the bank

5.2.2.9 changeLine()

```
void SystemModel::SystemModel::changeLine (
            uint8_t busNumber1,
            uint8_t busNumber2,
            double r,
            double x,
            double b )
```

Changes the parameters of the line between buses

Parameters

busNumber1	Ordinal number of the first bus
busNumber2	Ordinal number of the second bus
r	Series resistance of the transmission line PI equivalent
x	Series reactance of the transmission line PI equivalent
b	Shunt susceptance of the transmission line PI equivalent

5.2.2.10 changeTransformer()

```
void SystemModel::SystemModel::changeTransformer (
            uint8_t busNumber1,
            uint8_t busNumber2,
            double r,
            double x,
            double g,
            double b )
```

Changes the parameters of the transformer between buses

Parameters

busNumber1	0rdinal number of the first bus
busNumber2	0rdinal number of the second bus
r	Series resistance of the transformer PI equivalent
x	Series reactance of the transformer PI equivalent
g	Shunt conductance of the transformer PI equivalent
b	Shunt susceptance of the transformer PI equivalent

5.2.2.11 getAdmittanceMatrix()

```
AdmittanceMatrix SystemModel::SystemModel::getAdmittanceMatrix ( ) const  [inline]
```

5.2.2.12 getBranches()

```
std::vector< Branch > SystemModel::SystemModel::getBranches ( ) const  [inline]
```

5.2.2.13 getBus()

```
SystemModel::Bus & SystemModel::SystemModel::getBus (
             uint8_t busNumber )
```

Gets the bus with the given bus number

Parameters

busNumber	Ordinal number of the desired bus

Returns

Bus with the given bus number

5.2.2.14 getBusFunctions()

```
SystemModel::fi SystemModel::SystemModel::getBusFunctions (
             uint8_t busNumber ) const
```

Gets the bus functions (fi_P and fi_Q) for the desired bus

Parameters

busNumber	Ordinal number of the desired bus

Returns

Bus functions for the given bus in the form of std::pair of functions, where both functions have a std::vector of doubles as parameters and return a double

5.2.2.15 getCapacitorBanks()

```
std::vector< std::tuple< uint8_t, double, ThreePhaseLoadConfigurationsType > > SystemModel↩
::SystemModel::getCapacitorBanks ( ) const  [inline]
```

5.2.2.16 getDerivativesOfBusFunctions()

```
SystemModel::dfidx SystemModel::SystemModel::getDerivativesOfBusFunctions (
             uint8_t busNumber ) const
```

Gets the derivates of the bus functions (dfi_P/dx and dfi_Q/dx) for the desired bus (two rows of the Jacobian associated with the given bus functions)

Parameters

busNumber	Ordinal number of the desired bus

Returns

Derivatives of the bus functions for the given bus in the form of std::pair of std::vector-s of functions, where both functions have a std::vector of doubles as parameters and return a double

5.2.2.17 getNumberOfBuses()

```
uint8_t SystemModel::SystemModel::getNumberOfBuses ( ) const  [inline]
```

5.2.2.18 hasSlackBeenAssigned()

```
bool SystemModel::SystemModel::hasSlackBeenAssigned ( ) const
```

Check whether the slack bus has been assigned

Returns

True if the slack bus has been assigned and false otherwise

5.2.2.19 removeBranch()

```
void SystemModel::SystemModel::removeBranch (
            uint8_t busNumber1,
            uint8_t busNumber2 )
```

Removes a line or transformer between buses

Parameters

busNumber1	0rdinal number of the first bus
busNumber2	0rdinal number of the second bus

5.2.2.20 removeBus()

```
void SystemModel::SystemModel::removeBus (
            uint8_t busNumber )
```

Removes the given bus from the system

Parameters

busNumber	Ordinal number of the desired bus

5.2.2.21 removeCapacitorBank()

```
void SystemModel::SystemModel::removeCapacitorBank (
            uint8_t busNumber )
```

Removes the capacitor bank that is connected to the given bus

Parameters

busNumber	Ordinal number of the desired bus

5.2.3 Friends And Related Function Documentation

5.2.3.1 operator<<

```
std::ostream & operator<< (
            std::ostream & stream,
            const SystemModel & systemModel )  [friend]
```

The documentation for this class was generated from the following files:

- systemModel.h
- systemModel.cpp

Chapter 6

File Documentation

6.1 export.cpp File Reference

```
#include "export.h"
#include "systemModel.h"
#include <algorithm>
#include <iostream>
#include <fstream>
#include <tuple>
```

Functions

- void exportToLatex (SystemModel::SystemModel s)

  Exports SystemModel to the main.tex file
- void exportToHTML (SystemModel::SystemModel s)

  Exports SystemModel to the main.html file
- void exportToTxt (const char ∗filename, SystemModel::SystemModel s)

  Exports SystemModel to the .tex file

Variables

- const double eps { 1e-10 }

6.1.1 Function Documentation

6.1.1.1 exportToHTML()

```
void exportToHTML (
          SystemModel::SystemModel s )
```

Exports SystemModel to the main.html file

Parameters

SystemModel::SystemModel	System model

Returns

6.1.1.2 exportToLatex()

```
void exportToLatex (
            SystemModel::SystemModel s )
```

Exports SystemModel to the main.tex file

Parameters

SystemModel::SystemModel	System model

Returns

6.1.1.3 exportToTxt()

```
void exportToTxt (
            const char * filename,
            SystemModel::SystemModel s )
```

Exports SystemModel to the .tex file

Parameters

const char∗	Name of the file
SystemModel::SystemModel	System model

Returns

6.1.2 Variable Documentation

6.1.2.1 eps

```
const double eps { 1e-10 }
```

6.2 export.h File Reference

```
#include <fstream>
#include <iostream>
#include "systemModel.h"
```

Functions

- void exportToLatex (SystemModel::SystemModel s)

  Exports SystemModel to the main.tex file
- void exportToHTML (SystemModel::SystemModel s)

  Exports SystemModel to the main.html file
- void exportToTxt (const char ∗filename, SystemModel::SystemModel s)

  Exports SystemModel to the .tex file

6.2.1 Function Documentation

6.2.1.1 exportToHTML()

```
void exportToHTML (
            SystemModel::SystemModel s )
```

Exports SystemModel to the main.html file

Parameters

SystemModel::SystemModel	System model

Returns

6.2.1.2 exportToLatex()

```
void exportToLatex (
            SystemModel::SystemModel s )
```

Exports SystemModel to the main.tex file

Parameters

SystemModel::SystemModel	System model

Returns

6.2.1.3 exportToTxt()

```
void exportToTxt (
             const char * filename,
             SystemModel::SystemModel s )
```

Exports SystemModel to the .tex file

Parameters

const char∗	Name of the file
SystemModel::SystemModel	System model

Returns

6.3 export.h

Go to the documentation of this file.

```
#pragma once
#include <fstream>
#include <iostream>
#include "systemModel.h"

void exportToLatex(SystemModel::SystemModel s);

void exportToHTML(SystemModel::SystemModel s);

void exportToTxt(const char* filename, SystemModel::SystemModel s);
```

6.4 import.cpp File Reference

```
#include <algorithm>
#include <iostream>
#include <fstream>
#include <tuple>
#include "systemModel.h"
```

Functions

- void importFromTxt (const char ∗filename, SystemModel::SystemModel &systemModel)

  Imports SystemModel from the .txt file

6.4.1 Function Documentation

6.4.1.1 importFromTxt()

```
void importFromTxt (
            const char * filename,
            SystemModel::SystemModel & systemModel )
```

Imports SystemModel from the .txt file

Parameters

const char∗	Name of the file
SystemModel::SystemModel	System model

Returns

6.5 import.h File Reference

```
#include "systemModel.h"
```

Functions

- void importFromTxt (const char ∗filename, SystemModel::SystemModel &s)

  Imports SystemModel from the .txt file

6.5.1 Function Documentation

6.5.1.1 importFromTxt()

```
void importFromTxt (
            const char * filename,
            SystemModel::SystemModel & systemModel )
```

Imports SystemModel from the .txt file

Parameters

const char∗	Name of the file
SystemModel::SystemModel	System model

Returns

6.6 import.h

Go to the documentation of this file.

```
#pragma once

#include "systemModel.h"


void importFromTxt(const char* filename, SystemModel::SystemModel& s);
```

6.7 main.cpp File Reference

```
#include <iostream>
#include "systemModel.h"
#include "newtonRaphson.h"
#include "export.h"
#include "import.h"
```

Functions

- int main ()

Variables

- const double eps { 1e-10 }

6.7.1 Function Documentation

6.7.1.1 main()

```
int main ( )
```

6.7.2 Variable Documentation

6.7.2.1 eps

```
const double eps { 1e-10 }
```

6.8 newtonRaphson.cpp File Reference

```
#include "systemModel.h"
#include "newtonRaphson.h"
#include <math.h>
#include <algorithm>
#include <iostream>
#include <vector>
```

Functions

- template<typename T >
  std::vector< T > operator∗ (const std::vector< std::vector< T > > &matrix, const std::vector< T > &vector)

  Matrix and vector product operator overload
- template<typename T >
  std::vector< std::vector< T > > operator- (const std::vector< std::vector< T > > &matrix)

  Unary minus sign matrix operator overload
- std::vector< double > absVector (const std::vector< double > &vec)

  Absolute value of vector
- void cofactor (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &t, int p, int q, int n)

  Cofactor of the matrix
- double determinant (std::vector< std::vector< double > > matrix, int n)

  Determinant of matrix
- void adjoint (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &adj)

  Adjoint matrix
- std::vector< std::vector< double > > inverseMatrix (const std::vector< std::vector< double > > &matrix, double eps=1e-10)

  Inverse matrix
- int newtonRaphson (SystemModel::SystemModel sm, int maxNumberOfIter, double eps, std::vector< double > x0, std::vector< double > &x, double &err, int &iter)

  Newton Raphson method

6.8.1 Function Documentation

6.8.1.1 absVector()

```
std::vector< double > absVector (
            const std::vector< double > & vec )
```

Absolute value of vector

Parameters

std::vector<double>	Vector of double elements

Returns

Absolute value of elements in the argument vector

6.8.1.2 adjoint()

```
void adjoint (
            const std::vector< std::vector< double > > & matrix,
            std::vector< std::vector< double > > & adj )
```

Adjoint matrix

Parameters

std::vector<std::vector<double>>	Matrix to get adjoint from
std::vector<std::vector<double>>	Referece to adjoint matrix

Returns

6.8.1.3 cofactor()

```
void cofactor (
            const std::vector< std::vector< double > > & matrix,
            std::vector< std::vector< double > > & t,
            int p,
            int q,
            int n )
```

Cofactor of the matrix

Parameters

std::vector<std::vector<double>>	Matrix to get cofactor from
std::vector<std::vector<double>>	Cofactor matrix
int	Row of the cofactor that needs to be found
int	Column of the cofactor that needs to be found
int	Size of square matrix

Returns

6.8.1.4 determinant()

```
double determinant (
            std::vector< std::vector< double > > matrix,
            int n )
```

Determinant of matrix

Parameters

std::vector<std::vector<double>>	Matrix to get determinant from
int	Size of square matrix

Returns

Double value of determinant

6.8.1.5 inverseMatrix()

```
std::vector< std::vector< double > > inverseMatrix (
            const std::vector< std::vector< double > > & matrix,
            double eps = 1e-10 )
```

Inverse matrix

Parameters

std::vector<std::vector<double>>	Matrix to get inverse from
double	Sinuglarity check

Returns

Matrix that is inverse from the first argument

6.8.1.6 newtonRaphson()

```
int newtonRaphson (
            SystemModel::SystemModel sm,
            int maxNumberOfIter,
            double eps,
            std::vector< double > x0,
            std::vector< double > & x,
            double & err,
            int & iter )
```

Newton Raphson method

Parameters

SystemModel::SystemModel	System model
int	Maximum number of iterations
double	Maximum tolerance
std::vector<double>	Starting solution vector
std::vector<double>	Reference to solution vector
double	Reference to tolerance achieved
int	Reference to number of iterations preformed

Returns

Int value that shows if the system converges or not. Returns 1 if converges, returns 0 if it does not

6.8.1.7 operator*()

```
template<typename T >
std::vector< T > operator* (
             const std::vector< std::vector< T > > & matrix,
             const std::vector< T > & vector )
```

Matrix and vector product operator overload

Parameters

std::vector<std::vector<T>>	Vector of vector type
std::vector<T>	Vector type

Returns

Vector that is the result of matrix and vector product

6.8.1.8 operator-()

```
template<typename T >
std::vector< std::vector< T > > operator- (
             const std::vector< std::vector< T > > & matrix )
```

Unary minus sign matrix operator overload

Parameters

std::vector<std::vector<T>>	Matrix of type

Returns

Reverse sign elements of matrix

6.9 newtonRaphson.h File Reference

```
#include "systemModel.h"
#include <vector>
#include <math.h>
#include <algorithm>
```

Functions

- template<typename T >
  std::vector< T > operator∗ (const std::vector< std::vector< T > > &matrix, const std::vector< T > &vector)

  Matrix and vector product operator overload
- template<typename T >
  std::vector< std::vector< T > > operator- (const std::vector< std::vector< T > > &matrix)

  Unary minus sign matrix operator overload
- std::vector< double > absVector (const std::vector< double > &vec)

  Absolute value of vector
- void cofactor (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &t, int p, int q, int n)

  Cofactor of the matrix
- double determinant (std::vector< std::vector< double > > matrix, int n)

  Determinant of matrix
- void adjoint (const std::vector< std::vector< double > > &matrix, std::vector< std::vector< double > > &adj)

  Adjoint matrix
- std::vector< std::vector< double > > inverseMatrix (const std::vector< std::vector< double > > &matrix, double eps)

  Inverse matrix
- int newtonRaphson (SystemModel::SystemModel sm, int maxNumberOfIter, double eps, std::vector< double > x0, std::vector< double > &x, double &err, int &iter)

  Newton Raphson method

6.9.1 Function Documentation

6.9.1.1 absVector()

```
std::vector< double > absVector (
          const std::vector< double > & vec )
```

Absolute value of vector

Parameters

std::vector<double>	Vector of double elements

Returns

Absolute value of elements in the argument vector

6.9.1.2 adjoint()

```
void adjoint (
            const std::vector< std::vector< double > > & matrix,
            std::vector< std::vector< double > > & adj )
```

Adjoint matrix

Parameters

std::vector<std::vector<double>>	Matrix to get adjoint from
std::vector<std::vector<double>>	Referece to adjoint matrix

Returns

6.9.1.3 cofactor()

```
void cofactor (
            const std::vector< std::vector< double > > & matrix,
            std::vector< std::vector< double > > & t,
            int p,
            int q,
            int n )
```

Cofactor of the matrix

Parameters

std::vector<std::vector<double>>	Matrix to get cofactor from
std::vector<std::vector<double>>	Cofactor matrix
int	Row of the cofactor that needs to be found
int	Column of the cofactor that needs to be found
int	Size of square matrix

Returns

6.9.1.4 determinant()

```
double determinant (
            std::vector< std::vector< double > > matrix,
            int n )
```

Determinant of matrix

Parameters

std::vector<std::vector<double>>	Matrix to get determinant from
int	Size of square matrix

Returns

Double value of determinant

6.9.1.5 inverseMatrix()

```
std::vector< std::vector< double > > inverseMatrix (
            const std::vector< std::vector< double > > & matrix,
            double eps = 1e-10 )
```

Inverse matrix

Parameters

std::vector<std::vector<double>>	Matrix to get inverse from
double	Sinuglarity check

Returns

Matrix that is inverse from the first argument

6.9.1.6 newtonRaphson()

```
int newtonRaphson (
            SystemModel::SystemModel sm,
            int maxNumberOfIter,
            double eps,
            std::vector< double > x0,
            std::vector< double > & x,
            double & err,
            int & iter )
```

Newton Raphson method

Parameters

SystemModel::SystemModel	System model
int	Maximum number of iterations
double	Maximum tolerance
std::vector<double>	Starting solution vector
std::vector<double>	Reference to solution vector
double	Reference to tolerance achieved
int	Reference to number of iterations preformed

Returns

Int value that shows if the system converges or not. Returns 1 if converges, returns 0 if it does not

6.9.1.7 operator∗()

```
template<typename T >
std::vector< T > operator* (
             const std::vector< std::vector< T > > & matrix,
             const std::vector< T > & vector )
```

Matrix and vector product operator overload

Parameters

std::vector<std::vector<T>>	Vector of vector type
std::vector<T>	Vector type

Returns

Vector that is the result of matrix and vector product

6.9.1.8 operator-()

```
template<typename T >
std::vector< std::vector< T > > operator- (
             const std::vector< std::vector< T > > & matrix )
```

Unary minus sign matrix operator overload

Parameters

std::vector<std::vector<T>>	Matrix of type

Returns

Reverse sign elements of matrix

6.10 newtonRaphson.h

Go to the documentation of this file.

```
#pragma once
#include "systemModel.h"
#include <vector>
#include <math.h>
#include <algorithm>

template <typename T>
std::vector<T> operator *(const std::vector<std::vector<T»& matrix, const std::vector<T>& vector);

template <typename T>
std::vector<std::vector<T» operator -(const std::vector<std::vector<T»& matrix);

std::vector<double> absVector(const std::vector<double>& vec);

void cofactor(const std::vector<std::vector<double»& matrix, std::vector<std::vector<double»& t, int p,
    int q, int n);

double determinant(std::vector<std::vector<double» matrix, int n);

void adjoint(const std::vector<std::vector<double»& matrix, std::vector<std::vector<double»& adj);

std::vector<std::vector<double» inverseMatrix(const std::vector<std::vector<double»& matrix, double eps);

int newtonRaphson(SystemModel::SystemModel sm, int maxNumberOfIter, double eps, std::vector<double> x0,
    std::vector<double>& x, double& err, int& iter);
```

6.11 systemModel.cpp File Reference

```
#include "systemModel.h"
#include <stdexcept>
#include <algorithm>
#include <iomanip>
#include <cmath>
```

Macros

- #define PI 4 ∗ std::atan(1.0)

6.11.1 Macro Definition Documentation

6.11.1.1 PI

```
#define PI 4 ∗ std::atan(1.0)
```

6.12 systemModel.h File Reference

```
#include <vector>
#include <complex>
#include <optional>
#include <functional>
#include <utility>
#include <tuple>
```

Classes

- class SystemModel::Bus
- class SystemModel::SystemModel

Namespaces

- namespace SystemModel

Typedefs

- using SystemModel::fi = std::pair< std::function< double(std::vector< double >)>, std::function< double(std::vector< double >)> >
- using SystemModel::dfidx = std::pair< std::vector< std::function< double(std::vector< double >)> >, std↵ ::vector< std::function< double(std::vector< double >)> > >
- using SystemModel::AdmittanceMatrix = std::vector< std::tuple< uint8_t, uint8_t, std::complex< double > > >
- using SystemModel::Branch = std::tuple< TypeOfBranch, uint8_t, uint8_t, double, double, double, double >

Enumerations

- enum class SystemModel::TypeOfBus { SystemModel::Slack , SystemModel::PV , SystemModel::PQ }
- enum class SystemModel::ThreePhaseLoadConfigurationsType { SystemModel::Star , SystemModel::GroundedStar , SystemModel::Delta }
- enum class SystemModel::TypeOfBranch { SystemModel::Line , SystemModel::Transformer }

Functions

- std::ostream & SystemModel::operator<< (std::ostream &stream, const SystemModel &systemModel)

  Output stream operator overload

6.13 systemModel.h

Go to the documentation of this file.

```
#pragma once
#include <vector>
#include <complex>
#include <optional>
#include <functional>
#include <utility>
#include <tuple>



namespace SystemModel {
    enum class TypeOfBus { Slack, PV, PQ };



    enum class ThreePhaseLoadConfigurationsType { Star, GroundedStar, Delta };



    enum class TypeOfBranch { Line, Transformer };



    using fi = std::pair<std::function<double(std::vector<double>)>,
    std::function<double(std::vector<double>)»;



    using dfidx = std::pair<std::vector<std::function<double(std::vector<double>)»,
    std::vector<std::function<double(std::vector<double>)»>;



    using AdmittanceMatrix = std::vector<std::tuple<uint8_t, uint8_t, std::complex<double»>;



    using Branch = std::tuple<TypeOfBranch, uint8_t, uint8_t, double, double, double, double>;



    class Bus {
        TypeOfBus typeOfBus;
        std::optional<double> voltageMagnitude;
        std::optional<double> voltagePhase;
        std::optional<double> activePower;
        std::optional<double> reactivePower;
    public:
        Bus(TypeOfBus typeOfBus) : typeOfBus{ typeOfBus } {}

        TypeOfBus getTypeOfBus() const {
            return typeOfBus;
        }

        void setVoltageMagnitude(double voltageMagnitude);

        void setVoltagePhase(double voltagePhase);

        void setActivePower(double activePower);

        void setReactivePower(double reactivePower);

        std::optional<double> getVoltageMagnitude() const;

        std::optional<double> getVoltagePhase() const;

        std::optional<double> getActivePower() const;

        std::optional<double> getReactivePower() const;
    };



    class SystemModel {
        AdmittanceMatrix admittanceMatrix;
        uint8_t numberOfBuses{};
        std::vector<Bus> buses;
        const uint8_t maxNumberOfBuses;
        bool checkForConnectionBetweenToBuses(uint8_t busNumber1, uint8_t busNumber2) const;
        std::vector<Branch> branches;
        void addBranchToAdmittanceMatrix(uint8_t busNumber1, uint8_t busNumber2, double r, double x,
    double g, double b);
    std::vector<std::tuple<uint8_t, double, ThreePhaseLoadConfigurationsType» capacitorBanks;
    void addCapacitorBankToAdmittanceMatrix(uint8_t busNumber, double b, ThreePhaseLoadConfigurationsType configurationType);
    void recalculateAdmittanceMatrix();
public:
    SystemModel(uint8_t maxNumberOfBuses) : maxNumberOfBuses{ maxNumberOfBuses } {}

    AdmittanceMatrix getAdmittanceMatrix() const {
        return admittanceMatrix;
    }

    uint8_t getNumberOfBuses() const {
        return numberOfBuses;
    }

    Bus& getBus(uint8_t busNumber);

    void addBus(TypeOfBus typeOfBus);

    void addLoad(uint8_t busNumber, double activePower, double reactivePower);

    void addLine(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b);

    friend std::ostream& operator «(std::ostream& stream, const SystemModel& systemModel);

    void addGenerator(uint8_t busNumber, double voltageMagnitude, double activePower);

    void addSlackGenerator(uint8_t busNumber, double voltageMagnitude, double voltagePhase);

    bool hasSlackBeenAssigned() const;

    void addTransformer(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g, double b);

    void addCapacitorBank(uint8_t busNumber, double b, ThreePhaseLoadConfigurationsType configurationType);

    fi getBusFunctions(uint8_t busNumber) const;

    dfidx getDerivativesOfBusFunctions(uint8_t busNumber) const;

    void removeBranch(uint8_t busNumber1, uint8_t busNumber2);

    void changeLine(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double b);

    void changeTransformer(uint8_t busNumber1, uint8_t busNumber2, double r, double x, double g, double b);

    std::vector<Branch> getBranches() const {
        return branches;
    }

    void removeBus(uint8_t busNumber);

    void removeCapacitorBank(uint8_t busNumber);

    void changeCapacitorBank(uint8_t busNumber, double b, ThreePhaseLoadConfigurationsType configurationType);

    std::vector<std::tuple<uint8_t, double, ThreePhaseLoadConfigurationsType» getCapacitorBanks() const {
        return capacitorBanks;
    }
};



std::ostream& operator «(std::ostream& stream, const SystemModel& systemModel);
}
```

Index